Transforms of the Computerized Patient Record Data Model: From Transactions to Analytical Processing

Kip Canfield, PhD, Marcelo Silva, MS,
Sunny Zhaohui, BS, Roopak Manchanda, MS, Patti Abbott, RN MS
Laboratory for Healthcare Informatics
Department of Information Systems
University of Maryland, UMBC

ABSTRACT

The design of Computerized Patient Record databases needs to be different depending on whether the focus of the database is transaction processing or analytical processing. Transaction processing includes the operational use of databases for single patient update during clinical encounters. Analytical processing includes cross-patient querying for research or clinical management. These different design needs are defined. A case study is presented for the methodology that transforms transaction databases to analytical ones. The resulting analytical design is argued to be successfully evaluated according to specific criteria of useability and maintainability. This methodology is easily replicated in any relational database environment.

INTRODUCTION

Computerized Patient Records (CPRs) are used primarily for reading and writing information about individual patients. A transaction system is required to do this. Patient record databases are updated with new information during clinical encounters. This requires strict normalization and referential integrity in order to assure valid updates. Other more analytical uses of patient records do not require these to hold. In fact, analytical processing is hampered by the strict structures of transaction databases. There has been a lot of activity in the database literature lately about OLAP (On-Line Analytical Processing) and Data Warehousing [1-3].

This paper uses a case study of the transformation of a transaction-oriented CPR into an analytically-oriented system. This case involves an operational CPR in a geriatrics clinic. This system is designed according to the standard techniques of relational database management systems such as normalization. The transformed analytical system is a read-only database whose design is influenced by the emerging OLAP literature. This transformed system is evaluated according to two parameters:

- Useability
- Maintainability

Useability is defined as the ability of the OLAP design to meet the query needs of users. The system must give access to the needed information and be usable by intended users. The performance of the OLAP system must also allow faster retrieval times for cross-patient queries. Maintainability issues include database administrator (DBA) tasks. The creation, update, and maintenance of the OLAP system must be within the typical skill set of a database administrator and not take undue time or attention. The remainder of this paper describes the methodology for transforming a transaction-oriented CPR design to an OLAP design. This design is then evaluated according to its useability and maintainability.

METHODS

The CPR used in this study has been the primary patient record at a research-oriented geriatrics clinic in Baltimore for about 3 years. It contains about 200 megabytes of patient information. The users of this CPR do both clinical documentation of patients during encounters and analytical processing of cross-patient information to create research data sets.

The CPR contains all information on the patients in the research clinic. This information comes from several sources, not all directly entered from the CPR data-entry interface. For example, all lab data comes from the hospital information system through a specially developed interface. Additionally, special interfaces have been developed for psychometric data. Review of systems data and other encounter-captured patient data is entered through the graphical interface of the CPR called GERI. Gupta's SQLWindows is used as the application development environment for GERI. The information flow for the clinic is shown in Figure 1. The transaction repository is a client server database where client application programs submit updates to the database server. This type of transaction-oriented repository has sometimes been called the Operational Data Store (ODS)[2].

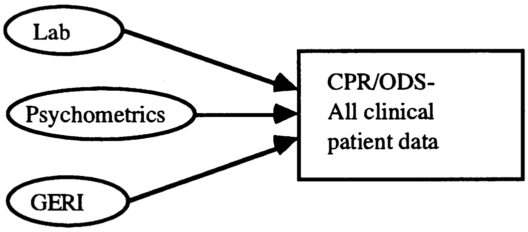

Figure 1. The Operational Data Store.

The data model of the transaction CPR is shown in Figure 2. It features the traditional design requirements of normalization and referential integrity constraints. This assures consistent updates during clinical encounters. In the discussion below entities refer to the model and tables refer to the database implementation. All models are assumed to be at the level of implementation and so there is no practical difference here.

This model captures the data from clinical encounters in the left-hand side of the diagram where each *Intervention* is a clinical event (either an action or an observation) and it is linked to information about the encounter, the provider, and the current active patient problems. The upper right-hand quadrant of the diagram shows the tables that make up the data dictionary for the Controlled Medical Terminology (CMT). The lower right-hand quadrant of the diagram shows the data dictionary for the protocols used in the research clinic. It seems clear that this abundance of normalized entities would present a formidable obstacle to cross-patient queries by the researchers. The single patient views for CPR update have been programmed into the client application GERI at relatively great expense. It is even more difficult to program such views for analytical processing because the requirements are more fluid.

The methodology discussed below makes a formal transform of the transaction model above to produce the analytical model. The transform is said to be formal because it is a sequence of SQL commands which are based on relational algebra. The steps in the methodology are:

1. Use a SQL script to create a single table for all patient information.
2. Use a SQL script to create selector tables.
3. Use a SQL script to create aggregation tables.
4. Update the meta-information table.
5. Create the end-user application with off-the-shelf tools.

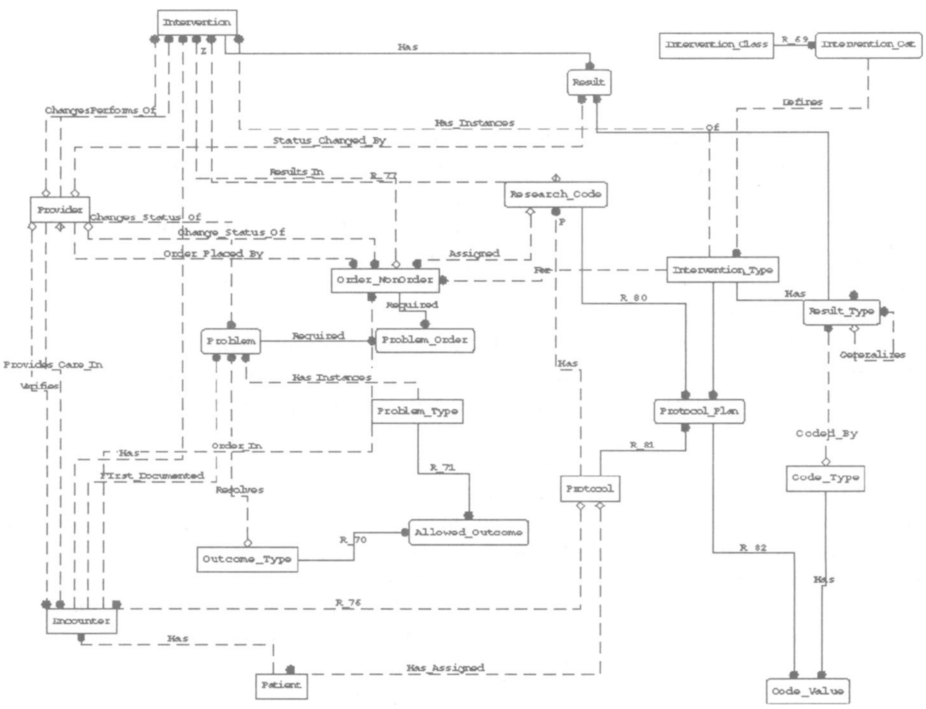

Figure 2. The Transaction Data Model.

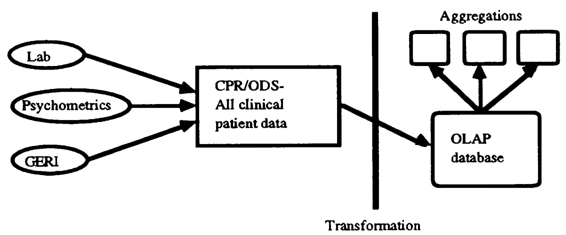

Figure 3. Transformation of a Transaction Database for OLAP.

When these steps are carried out, the resulting analytical database is used as a read-only database for cross-patient queries. This addition to the information flow is shown in Figure 3. The resulting database tables are populated with redundant relations that would result in multiple update and deletion anomalies if used as a transaction database.

Step 1 requires a relational join across all tables that include patient information of interest. There can be different relations generated depending on the interest of the OLAP database users. For example, selected attributes from the following entities were used for this project: *Patient*, *Encounter*, *Provider*, *Problem*, *Intervention*, and *Result*. This large join results in a relation that is written to a new table in the OLAP database called *Patient_data*. This is accomplished with a script that is a series of consecutively executed SQL statements. All relational database server products support such scripts. Gupta's SQLBase is used as the database server for this project.

Step 2 creates the selector tables/entities that are used to select information from *Patient_data*. Some of these tables are created by extracting relations from the ODS and some are created specifically for the OLAP database. For example, the CMT data dictionary information is extracted with a join to form a large table. Selected columns from the following entities were used for this project: *Event_type*, *Result_type*, *Code_type*, *Code_value*, *Intervention_class*, *Intervention_cat*, and *Intervention_type*. This creates a large redundant relation that allows all elements for the CMT to be selected from a single table. A similar process was used to extract the protocol information. All these extractions are performed with SQL scripts. Other relations/tables are created (new) to allow selection of specific groups of patient records. In this project, certain subgroups of patients assigned to protocols were tracked by researchers. These groups are defined by separate relations according to patient data values. For example, certain patient groups were tracked by date ranges rather than strictly by assigned protocol. All selector tables allow the relevant patient data to be selected with one join. This is sometimes referred to as a star structure in the OLAP literature [4] which supports a conceptually multidimensional database design in a relational database environment.

Step 3 uses SQL scripts to extract from the OLAP database. These scripts aggregate the patient data to a more compact form for monitoring and reporting cross-patients. For example, summations or averages of values such as total visits/week or average weight/protocol can be saved in special aggregation tables.

Step 4 updates a special table with meta-information about the OLAP database. This information includes a date/time-stamped log of each script run. This information allows any update of the OLAP database to be repeated for data quality purposes and incremental updates of the OLAP tables. It also acts as a menu of the available tables in the end-user application.

Step 5 uses the resulting tables from steps 1-4 to create an application. This application is created with off-the-shelf database access tools that allow end-user graphical query capability and simple scripting. Intersolv's Q+E product is used in this project (many other similar tools are available). The simplicity of the star design given above allows these applications to be very simple to develop and use.

RESULTS

The OLAP database that results from the methods in this paper was evaluated according to specific questions under the general categories of useability and maintainability. The informal evaluation for this project used a special study group of 7 people. Three of these people were the authors of this paper. The other four people include 1 DBA and 3 end-users. These group members were all already users of the transaction database. The design and use of the OLAP database were explained to and discussed with this group. All evaluation questions were then addressed with the group. The results are given below. Each is discussed since the issues are complex and the number of participants is small.

Useability includes mostly user-oriented issues. The useability questions are:

- Can typical end-users use the application?

The end-users in the study group were able to perform the basic tasks required to use the OLAP system. Since only one join is required for any selection, this can be pre-defined and there is no need for end-users to know SQL or even use the graphical specification of joins supplied in Q+E. Even the non-SQL graphical interfaces for joins in these tools are difficult for typical users. The users were in general excited about their increased access to data without an intermediary.

- Can end-users extract the information they need?

End-users were able to extract information from the OLAP database if it was the result of the defined joins. Some of the queries were more complex than this and were problematic. For example, the primitive interface given supports natural joins and outer joins. Users had difficulty choosing which join they wanted correctly. Most users just use the default natural joins. User specification of outer joins are a training issue for further study.

- Can end-users create selector tables?

End-users were easily trained to create and update simple independent selector tables to subset patient data. They could not create selector tables that are extractions from the ODS.

- Can end-users format the results of their queries?

End-users had difficulty using cross-tabulation tools to format the results of their queries. They had no trouble extracting the data from the query tool and transferring it to a spreadsheet.

Maintainability includes mostly database administrator-oriented (DBA) issues. The maintainability questions are:

- Can DBAs perform the steps of the methodology?

DBAs have no trouble understanding the extraction process.

- Do DBAs consider the OLAP tasks unwanted duties?

DBAs do not consider the tasks extra work, because the OLAP database gives end-users increased access to the research patient data and reduces the load on the DBA. In this setting, the DBA is also responsible for acting as an intermediary for researcher queries. It also takes a processing load off of the ODS which increases the performance of the ODS.

- Can the OLAP database be updated easily and accurately?

The meta-information table allows the DBA to track all additions to the OLAP database, but is a crude tool. It does not automate any administrative functions. It does contain the information to allow re-extraction of specific portions of the OLAP database or incremental appending to the OLAP database.

- Are all development tasks possible without extensive programming?

Only most tasks were possible with no programming or limited scripting. There were additional requirements at the end-user query interface and DBA tools that would require more extensive programming and application development.

DISCUSSION

This case study has developed a methodology for transforming a transaction-oriented CPR database to an OLAP-oriented database repository for clinical researchers. This resulting OLAP database was informally evaluated for issues bearing on useability and maintainability.

The results of this case study are encouraging but not quite as straightforward as originally conceived. The general goals of better user access and simple maintenance were at least partly achieved. The lesson here is tautologous: Complex things are really complex. There is no magic solution that makes underlying complex tasks absolutely simple.

The star structure approach to OLAP database seems promising for giving end-users better access to patient data for OLAP. It does not however, in simple form, allow a level of complex querying that users require. It also results in a simple environment for DBAs. It does not, however, provide the desired level of automation. Both of these shortfalls require more application development. The major motivating factor for this study was to create an environment that could be created and maintained simply and respond in almost real time to changing user requirements [5]. This goal was at least partially achieved. Further study is underway to more clearly specify these problem areas.

References

1. Inmon, W.H., *Building the Data Warehouse*. 1992, Boston MA: QED Technical Publishing Group.
2. Inmon, W.H. and R.D. Hackathorn, *Using the Data Warehouse*. 1994, New York, NY: John Wiley & Sons : QED Publication.
3. Gilbreath, R.E., *Health Care Data Repositories: Components and a Model.* Healthcare Information Management, 1995. **9**(1): p. 63-73.
4. Finkelstein, R., *MDD: Database Reaches the Next Dimension.* Database Programming and Design, 1995. **8**(4).
5. Zahedi, F., *Quality Information Systems*. 1995, Danvers, MA: boyd & fraser publishing co.